



Pliris- Review and New Capabilities

Joseph D. Kotulski
jkotul@sandia.gov
Dept. 1652



Outline

- **History**
- **Code Features**
- **Member Functions**
- **Future Work**



History

- Developed by David Womble and Bruce Hendrickson for the N-Cube
- Ported to Paragon – NX message passing
- Code change to use MPI Protocol



History

- Used as the core solver for BEM methods in Org. 1652
- Electrostatics/ Magnetostatics
 - Double Precision Real Matrix Entries
- Steady-state Maxwell's Equations
 - Complex Matrix Entries



Core Code Information

- Written in C
- LU Factorization with partial pivoting
 - $A = LU$
- Parallel Features
 - Uses torus-wrap decomposition to minimize communication volume (transparent to user)
 - BLAS level three for high data re-use (dgemm)
 - BLAS only external software



Code Usage

- Right Hand Sides available prior to factor/solve
 - Factor / forward solve done at the same time
 - Followed by a call to back solve
 - L not permuted



Code Usage

- Right Hand Side(s) available after factor before solve
 - Factor performed
 - L permuted – permutation vector created
 - Forward solve / back solve



User Inputs

- **Number of processors**
 - Parallel job initiation
- **Number of processors per row**
 - Sets processor decomposition
- **Matrix and Right Hand Side**
 - Block fill capability



Matrix Decomposition Algorithm

- No processor will have more than one row or column than any other processor
 - to minimize communication
 - Balance fill assuming all elements of the matrix have the same computational co
- Similar for Right Hand Side



Processor Decomposition

8 Processors 4 processors for a row

0	1	2	3
4	5	6	7

Matrix Distribution on 2 Processors / 2 processors for a row

	my_col = 0	my_col = 1
my_row = 0	$a_{11} \quad a_{12} \quad a_{13}$ $a_{21} \quad a_{22} \quad a_{23}$ $a_{31} \quad a_{32} \quad a_{33}$ $a_{41} \quad a_{42} \quad a_{43}$ $a_{51} \quad a_{52} \quad a_{53}$	$a_{14} \quad a_{15}$ $a_{24} \quad a_{25}$ $a_{34} \quad a_{35}$ $a_{44} \quad a_{45}$ $a_{54} \quad a_{55}$
Processor 0		
my_first_row = 1	my_first_col = 1	my_rows = 5 my_cols = 3
Processor 1		
my_first_row = 1	my_first_col = 4	my_rows = 5 my_cols = 2

Matrix Distribution on 2 Processors / 1 processor for a row

my_col = 0

my_row = 0	<table border="1"><tr><td>a_{11}</td><td>a_{12}</td><td>a_{13}</td><td>a_{14}</td><td>a_{15}</td></tr><tr><td>a_{21}</td><td>a_{22}</td><td>a_{23}</td><td>a_{24}</td><td>a_{25}</td></tr><tr><td>a_{31}</td><td>a_{32}</td><td>a_{33}</td><td>a_{34}</td><td>a_{35}</td></tr></table>	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
a_{11}	a_{12}	a_{13}	a_{14}	a_{15}												
a_{21}	a_{22}	a_{23}	a_{24}	a_{25}												
a_{31}	a_{32}	a_{33}	a_{34}	a_{35}												
my_row = 1	<table border="1"><tr><td>a_{41}</td><td>a_{42}</td><td>a_{43}</td><td>a_{44}</td><td>a_{45}</td></tr><tr><td>a_{51}</td><td>a_{52}</td><td>a_{53}</td><td>a_{54}</td><td>a_{55}</td></tr></table>	a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{51}	a_{52}	a_{53}	a_{54}	a_{55}					
a_{41}	a_{42}	a_{43}	a_{44}	a_{45}												
a_{51}	a_{52}	a_{53}	a_{54}	a_{55}												

Processor 0 ■

my_first_row = 1 my_first_col = 1 my_rows = 3 my_cols = 5

Processor 1 ■

my_first_row = 4 my_first_col = 1 my_rows = 2 my_cols = 5

Matrix Distribution on 4 Processors / 2 processors for a row

	my_col = 0	my_col = 1
my_row = 0	a_{11} a_{12} a_{13}	a_{14} a_{15}
my_row = 1	a_{21} a_{22} a_{23}	a_{24} a_{25}
	a_{31} a_{32} a_{33}	a_{34} a_{35}
	a_{41} a_{42} a_{43}	a_{44} a_{45}
	a_{51} a_{52} a_{53}	a_{54} a_{55}

Processor 0  my_first_row = 1 my_first_col = 1 my_rows = 3 my_cols = 3

Processor 1  my_first_row = 1 my_first_col = 4 my_rows = 3 my_cols = 2

Processor 2  my_first_row = 4 my_first_col = 1 my_rows = 2 my_cols = 3

Processor 3  my_first_row = 4 my_first_col = 4 my_rows = 2 my_cols = 2

$$\begin{array}{cc|cc}
 a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
 a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\
 a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\
 \hline
 a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\
 a_{51} & a_{52} & a_{53} & a_{54} & a_{55}
 \end{array} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

Processor 0 █ my_first_row =1 my_first_col =1 my_rows = 3 my_cols = 3 my_rhs=1

Processor 1 █ my_first_row =1 my_first_col =4 my_rows = 3 my_cols = 2 my_rhs=0

Processor 2 █ my_first_row =4 my_first_col =1 my_rows = 2 my_cols = 3 my_rhs=1

Processor 3 █ my_first_row =4 my_first_col =4 my_rows = 2 my_cols = 2 my_rhs=0



Member Functions

Pliris () *Pliris Default constructor*

int **SetRHS (Epetra_Object *B)**

int **SetMatrix(Epetra_Object *A)**

int **GetDistribution(...)**

int **FactorSolve (Epetra_Object *A,)**

int **FactorSolve (Epetra_Object *A,)**

int **Factor (Epetra_Object *A,)**

int **FactorSolve (Epetra_Object *A,)**



Epetra Objects

- **Epetra_Object**
 - **Epetra_Vector**
 - **Epetra_MultiVector**
 - **Epetra_SerialDenseVector**
 - **Epetra_SerialDenseMatrix**



Simple Example

```
// Enroll into MPI                                Epetra_Vector A(map);  
  
MPI_Init(&argc,&argv);  
Epetra_MpiComm comm(MPI_COMM_WORLD);      // Fill Matrix and RHS  
  
Pliris solver;  
  
solver.GetDistribution( &nprocs_row,           // End Fill  
                        &matrix_size,  
                        &nrhs,  
                        &my_rows,  
                        &my_cols,  
                        &my_first_row,  
                        &my_first_col,  
                        &my_rhs,  
                        &my_row,  
                        &my_col );  
  
// Define map                                     // Unpack Answers  
  
Epetra_Map map(num_global_length,num_my_length,   MPI_Finalize();  
                0, comm);
```



Package Status

- Regression test added
 - Random matrix build
- Part of full distribution of Trilinos



Future Work

- **Code Enhancement – Higher Efficiency**
- **Additional Regression Tests**
- **Documentation**
 - Online
 - SAND Report
- **Tpetra interface (Complex Double Precision)**